ICA for Blind Source Separation

Sept. 2020

Chenkuan Liu

1 Introduction

The purpose of this project is to use independent component analysis (ICA) to accomplish blind source separation. We use a provided 5 by 44000 matrix N as our source signals. In particular, each row of the matrix represents a four second clip which are sampled at 11025. To mix the signals, we create a random matrix A, which has m rows (m is arbitrary) and 5 columns, to multiply with our N. Then, to recover the source signals, we implement the independent component analysis algorithm, which involves maximum likelihood estimation, sigmoid function simulation, and gradient descent learning. After recovering, we will compare the recovered signals versus the original signals. Finally, we will compute the correlations between the recovered and the source signals to demonstrate the efficiency of our algorithm.

2 Method

The key step of our algorithm is gradient descent learning. After we initialize our first-guess unmixing matrix W, we will use gradient descent to update it as $W = W + \Delta W$. In particular, ΔW is computed as follows:

> We first calculate $Y = WX$ using our original matrix W, then calculate Z such that $z_{i,j} = (1 + e^{-y_{i,j}})^{-1}$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, t\}$ (t is the length of the signals, which equals 44000 here); after setting our learning rate η, we compute $\Delta W = \eta(tI + (1 - 2Z)Y^T)W$ to update our W.

In general, the implementation of the ICA algorithm is straightforward. We will update W until the algorithm reaches our maximum iteration number. However, because of the large size of our given source matrix (which has 44000 columns), the guesses and choices of our parameters become important. In particular, for learning rate η, if it is too large, ΔW will contain large values and our program will experience overflow after a few iterations; if it is too small, the update will be slow and will influence our running time and recovery efficiency. The number of iterations also has similar issues.

We will first experiment on the small dataset "icaTest.mat", in which A is an 3 by 3 matrix and N is an 3 by 40 matrix, to demonstrate the effect of our ICA algorithm. And then we will move to the larger audio dataset. The results and outputs are displayed in the following section.

3 Results

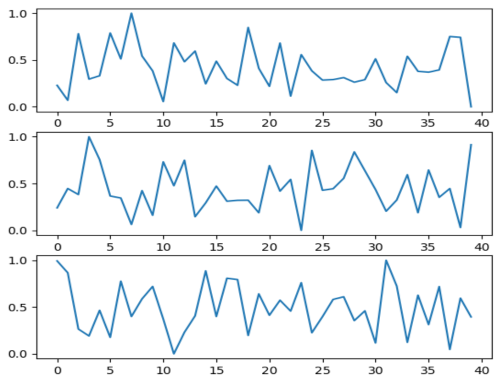
3.1 On Small Dataset

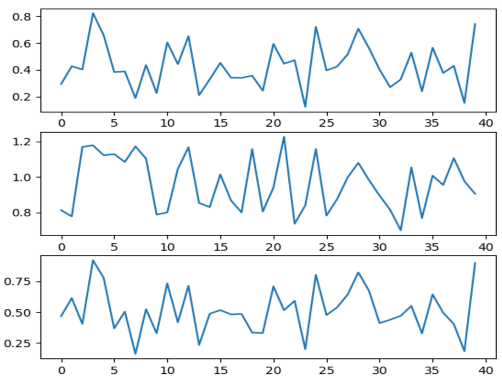
The initial guess of the unmixing matrix W is an 3 by 3 matrix with entries randomly and uniformly distributed from 0 to 0.1. We set our learning rate η to 0.01 and the number of iterations to 20000. After implementation, our program outputs the original signals, mixing signals, and recovered signals, and the correlation between original and recovered signals. Figure 1 below shows the plots of the signals, and Table 1 shows the correlation matrix.

original \ recovered	1	2	3
1	-0.52726043	0.98667858	-0.39634099
2	0.98838653	-0.35561275	-0.5701659
3	-0.4464573	-0.54398333	0.99118504

Table 1: Correlation Matrix of Small Dataset

We see that the first recovered signal corresponds to the second original signal, and the second recovered signal corresponds to the first original one. The recovered signals are almost identical to the original signals. The correlation matrix confirms the plots, with associated entries larger than 0.98. In general, it is a decent recovery.


(a) Original Signals


(b) Mixed Signals

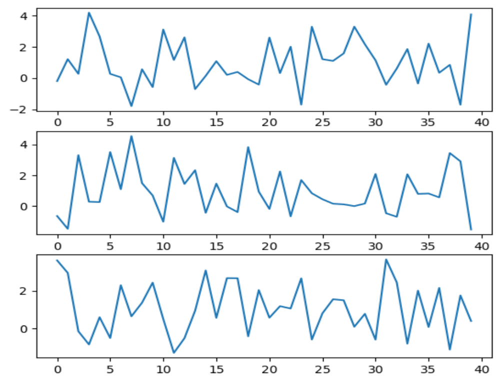

(c) Recovered Signals

Figure 1: Original, Mixed, and Recovered Signals of the Small Dataset

3.2 On Large Dataset

We now move to the large dataset, which is the main goal of the project. As the original matrix N here contains much more columns than the small dataset above, it is more possible to experience overflow or under-performance. And so, the choices of learning rate and number of iteration becomes far more crucial.

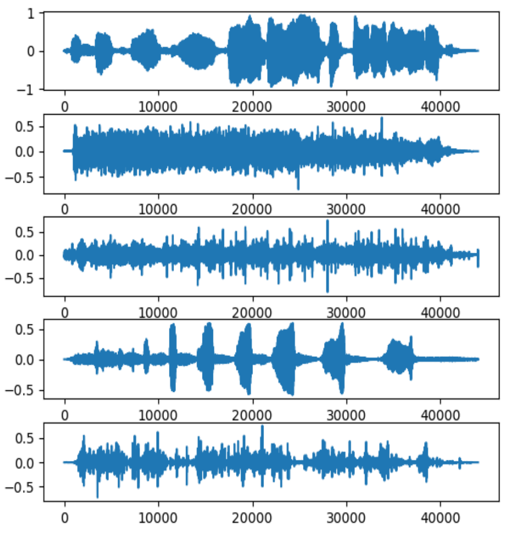
I set the mixing matrix A to be an 5 by 5 matrix with entries randomly and uniformly distributed from -5 to 5. The initial guess of unmixing matrix W is randomly and uniformly distributed from 0 to 0.001. After several trials, I set the learning rate η to 1×10^{-5} and the number of iteration to 100. Figure 2 on next page shows the plots of all the signals, and Table 2 shows the correlation matrix (I have rounded the values to 5 significant numbers for better readability).

recovered / original	1	2	3	4	5
1	3.1925e-03	9.9997e-01	3.4958e-03	7.3662e-03	2.9878e-03
2	1.4056e-03	1.3305e-03	9.0763e-03	-9.9996e-01	-2.7823e-04
3	2.5862e-03	7.1958e-04	-9.9991e-01	-8.9098e-03	-9.5324e-03
4	-2.0377e-03	4.1175e-03	1.0210e-02	-8.2985e-03	9.9994e-01
5	-9.9994e-01	-1.4067e-03	-1.2452e-02	-1.9618e-03	5.1251e-03

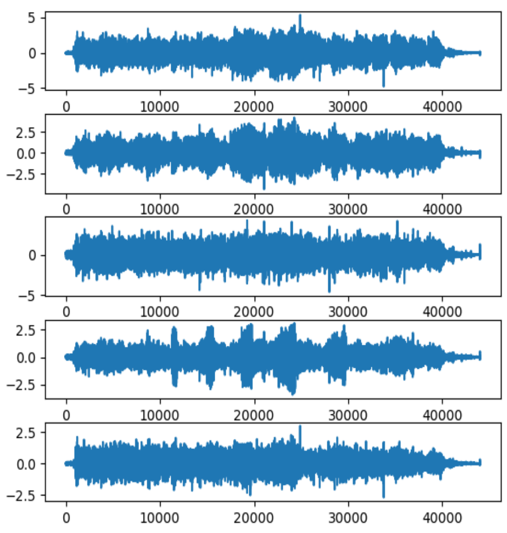
Table 2: Correlation Matrix of Large Dataset

Despite longer computation time, we see that the results are decent. The recovered signals are almost identical to the original ones, though the order becomes different and some signals are flipped with respect to the x-axis. The correlation matrix gives information on the correspondence between the original signals and recovered signals and whether the recovered signals are flipped or not. In particular, the correspondence at here is $1 \leftrightarrow 2$, $2 \leftrightarrow (-4)$, $3 \leftrightarrow (-3)$, $4 \leftrightarrow 5$, and $5 \leftrightarrow (-1)$, where the left number of the arrow denotes the order in original signals, the right number denotes the order in recovered signals, and negative sign denotes negative correlation.

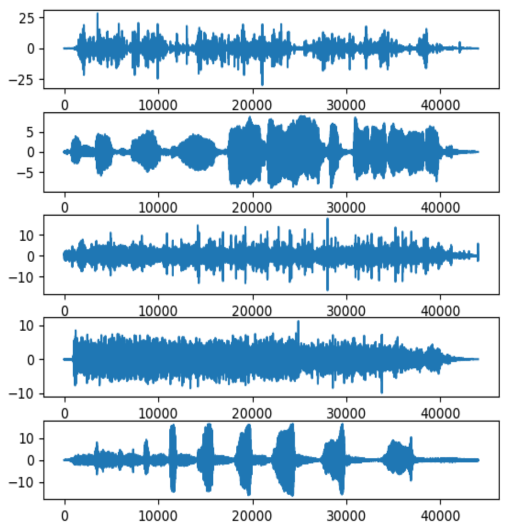
From the matrix, the correlation is either greater than 0.99 or smaller than -0.99, which means we almost achieved full recovery.



(a) Original Signals



(b) Mixed Signals



(c) Recovered Signals

Figure 2: Original, Mixed, and Recovered Signals of the Large Dataset

4 Summary

In general, the algorithm performs great on both small and large dataset. The correlation values between original and recovered signals in the large dataset are almost 1 or -1, which are even better than those in the small dataset. In addition, we also see that the learning rate and the iteration number are very important in the algorithm and we need to modify them base on specific situations. The small dataset has larger learning rate and far more iterations than those in the large dataset. If we still use $\eta = 0.01$ in large dataset, we will get overflow within a few iterations; if we use 20000 iterations in large dataset, it may take hours to finish. Therefore, within the program of the large dataset, I have set it to print the current estimation of W after each 20 iterations. In this way, it becomes easier to check the progress of the algorithm and modify the parameters.

5 References

- *numpy* library
- *matplotlib.pyplot* library
- *scipy.io* library: *loadmat*() for loading .mat files into Python, $wavfile.write$() for generating .wav sound files
- Andrew Ng's Lecture Notes:
  http://cs229.stanford.edu/notes2020spring/cs229-notes11.pdf